**MỞ ĐẦU**

Ngày nay, Trí tuệ nhân tạo đã và đang được ứng dụng nhiều vào thực tiễn. Trí tuệ nhân tạo đã chứng tỏ được thế mạnh của mình trong các công việc đòi hỏi khả năng suy nghĩ và tính toán tương tự như con người, trong đó, vấn đề tìm kiếm có đối thủ đang được áp dụng rất rộng rãi trong các trò chơi đối kháng. Xuất phát từ những vấn đề nêu trên, bài tập lớn “Phương pháp tìm kiếm có đối thủ. Mô tả cây trò chơi cho bài toán Dodgem. Sử dụng chiến lược cắt tỉa Alpha-Beta. Từ một trạng thái, hãy mô tả bằng chương trình cắt tỉa Beta trong phân tích độ sâu là 5” nhằm mục đích làm rõ hơn phương pháp tìm kiếm có đối thủ, mô tả cây trò chơi cho bài toán Dodgem và sử dụng chiến lược cắt tỉa Alpha- Beta để xây dựng chương trình minh họa.

Những phần trình bày chính trong bài tập lớn này bao gồm:

CHƯƠNG I. BÀI TOÁN TRÒ CHƠI DODGEM

CHƯƠNG II. XÂY DỰNG CHƯƠNG TRÌNH

**CHƯƠNG I. BÀI TOÁN TRÒ CHƠI DODGEM**

**I. MÔ TẢ BÀI TOÁN**

**1. Trò chơi tổng quát**

Xét các trò chơi giữa hai đối thủ là White (máy tính) và Black (con người). Hai “người” lần lượt thay phiên nhau thực hiện các nước đi bằng cách áp dụng một luật chơi cho trước xác định. Các luật này là như nhau đối với cả hai người chơi. Ta có các nhận xét sau đây:

– Các trò chơi như thế có thể biểu diễn trong một không gian trạng thái.

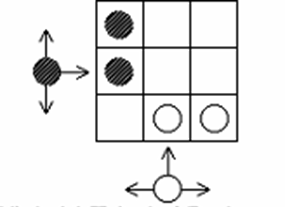
– Mỗi một tình huống (configuration) trong trò chơi biểu diễn một trạng thái.

– Các luật chơi tương ứng với các phép biến đổi trạng thái.

– Trạng thái bắt đầu trò chơi được cho trước. Trạng thái mà không thể có nước đi hợp lệ nào có thể thực hiện được gọi là trạng thái kết thúc. Có thể có một tập các trạng thái kết thúc.

**2. Trò chơi Dodgem**

Trò chơi Dogem được sáng tạo bởi Volin Vout (Berlek amp et al, 1982). Dodgem game là một chess game, giống như cờ Caro hay cờ tướng… Mỗi game đều có 2 người chơi. Một người là Min, người còn lại sẽ là Max. Hai người chơi sẽ thay phiên nhau đưa ra các nước đi của mình theo một quy luật nào đó. Trò chơi Dodgem có hai quân trắng và hai quân đen trên bàn cờ 3 x 3 như hình 1.



Hình 1. Trò chơi Dodgem

Luật chơi như sau:

– Quân đen: có thể đi lên trên, xuống dưới, hoặc sang phải; có thể đi ra ngoài bàn cờ (nếu ở cột ngoài cùng bên phải).

– Quân trắng: có thể đi sang trái, sang phải, hoặc lên trên; có thể đi ra ngoài bàn cờ (nếu ở hàng trên cùng của bàn cờ).

Trò chơi kết thúc khi 1 trong 2 người chơi đưa được hết các quân cờ của mình ra ngoài hoặc chặn không cho quân đối phương di chuyển (thì được coi là chiến thắng).

Như vậy, trò chơi Dodgem hoàn toàn phù hợp với mô hình bài toán trò chơi được mô tả tổng quát ở trên. Cụ thể là:

– Mỗi một nước đi tạo ra một bàn cờ mới với một thế cờ khác nhau, và đó chính là một trạng thái. Tập hợp tất cả các thế cờ có thể xảy ra chính là không gian trạng thái của trò chơi Dodgem.

– Một thế cờ (tình huống) là một trạng thái.

– Luật chơi cho từng bên White và Black như đã mô tả ở trên. Rõ ràng mỗi một nước đi áp dụng luật đã cho là một biến đổi trạng thái.

– Trạng thái bắt đầu được mô tả như hình vẽ.

– Tập hợp các trạng thái kết thúc, và do đó kết luận được ai thắng, ai thua như đã mô tả về điều kiện thắng, thua ở trên.

**II. GIẢI QUYẾT BÀI TOÁN**

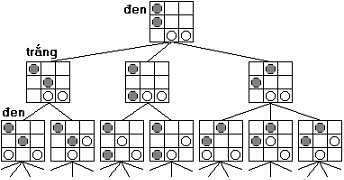
**1. Cây trò chơi**

Các trạng thái bàn cờ khác nhau trong quá trình chơi có thể biểu diễn thành một cây tìm kiếm (được gọi là cây trò chơi). Chúng ta tìm kiếm trên cây để tìm được nước đi tốt nhất. Cây trò chơi có các nút là các tình huống khác nhau của bàn cờ, các nhánh nối giữa các nút sẽ cho biết từ một tình huống bàn cờ chuyển sang tình huống khác thông qua chỉ một nước đi đơn nào đó. Tuy nhiên, các nước đi này diễn ra theo cặp do hai đấu thủ lần lượt tiến hành. Độ sâu của cây trò chơi là số tầng của cây. Cách xây dựng cây trò chơi:

– Gốc của cây ứng với trạng thái ban đầu.

– Gọi đỉnh ứng với trạng thái mà Trắng (Đen) đưa ra nước đi là đỉnh Trắng (Đen). Nếu một đỉnh là Trắng (Đen) ứng với trạng thái u, thì các đỉnh con của nó là tất cả các đỉnh biểu diễn trạng thái v (v nhận được từ u do Trắng (Đen) thực hiện nước đi hợp lệ). Do đó, trên cùng một mức của cây các đỉnh đều là Trắng hoặc đều là Đen, các lá của cây ứng với các trạng thái kết thúc.

Trong trò chơi Dodgem, giả sử Đen đi trước, ta có cây trò chơi được biểu diễn như trong hình 2.



Hình 2. Cây trò chơi Dodgem với Đen đi trước

**2. Hàm đánh giá**

Trên hình 2, để tìm nước đi tối ưu cho quân Đen, ta phải duyệt hết qua tất cả các đường đi có thể của nó và như thế ta sẽ tạo được cây trò chơi như trên.

Vậy làm sao để lựa chọn được nước đi thích hợp, đó chính là việc ta phải xây dựng Hàm đánh giá.

Hàm đánh giá Eval ứng với mỗi trạng thái u của trò chơi với một giá trị số Eval(u), giá trị này là sự đánh giá “độ lợi thế” của trạng thái u. Trạng thái u càng thuận lợi cho Trắng thì Eval(u) là số dương càng lớn; u càng thuận lợi cho Đen thì Eval(u) là số âm càng nhỏ; Eval(u) ≈ 0 đối với trạng thái không lợi thế cho ai cả.

Hàm Eval phải tính đến nhiều yếu tố: số quân còn lại trên bàn cờ, cách bố trí các quân trên bàn cờ,… Eval(u) = +∞ (dương vô cùng) nếu u là thế cờ thắng đối với Trắng và Eval(u) = -∞ (âm vô cùng) nếu u là thế cờ thắng đối với Đen.

Chú ý rằng hàm Eval là hàm đánh giá tĩnh (Static function). Nghĩa là nó chỉ đánh giá tính tối ưu tại một thế cờ cụ thể hiện đang xảy ra mà không xem xét được trong toàn bộ các thế cờ có thể xảy ra trong cuộc chơi.

Chất lượng của chương trình chơi cờ phụ thuộc rất nhiều vào hàm đánh giá. Nếu hàm đánh giá cho ta sự đánh giá không chính xác về các trạng thái, nó có thể hướng dẫn ta đi tới trạng thái được xem là tốt, nhưng thực tế lại rất bất lợi cho ta. Thiết kế một hàm đánh giá tốt là một việc khó, đòi hỏi ta phải quan tâm đến nhiều nhân tố: các quân còn lại của hai bên, sự bố trí của các quân đó,… ở đây có sự mâu thuẫn giữa độ chính xác của hàm đánh giá và thời gian tính của nó. Hàm đánh giá chính xác sẽ đòi hỏi rất nhiều thời gian tính toán, mà người chơi lại bị giới hạn bởi thời gian phải đưa ra nước đi.

Bây giờ ta đưa ra một cách đánh giá các trạng thái trong trò chơi Dodgem. Mỗi quân Trắng ở một vị trí trên bàn cờ được cho một giá trị tương ứng trong bảng hình 3a. Còn mỗi quân Đen ở một vị trí sẽ được cho một giá trị tương ứng trong bảng hình 3b:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 30 | 35 | 40 |  |  |  | -10 | -25 | -40 |  |
|  | 15 | 20 | 25 |  |  |  | -5 | -20 | -35 |  |
|  | 0 | 5 | 10 |  |  |  | 0 | -15 | -30 |  |
| a. Giá trị quân trắng | | | | |  | a. Giá trị quân đen | | | | |
| Hình 3. Đánh giá các quân trong trò chơi Dodgem | | | | | | | | | | |

Ngoài ra, nếu quân Trắng cản trực tiếp một quân Đen, nó được thêm 40 điểm, nếu cản gián tiếp nó được thêm 30 điểm (hình 4). Tương tự, nếu quân Đen cản trực tiếp quân Trắng nó được thêm -40 điểm, còn cản gián tiếp nó được thêm -30 điểm.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
| Hình 4. Đánh giá sự tương quan quân Trắng và Đen | | | | | | | | | | |

Áp dụng các qui tắc trên, ta tính được giá trị của trạng thái ở hình 5a là 75, giá trị của trạng thái hình 5b là -5.

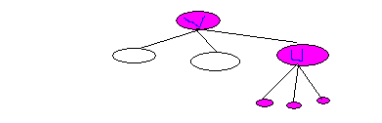
|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
| Hình 5. Giá trị của một số trạng thái trong trò chơi Dodgem | | | | | | | | | | |

Trong đánh giá trên, ta đã xét đến vị trí của các quân và mối tương quan giữa các quân. Một cách đơn giản để hạn chế không gian tìm kiếm là, khi cần xác định nước đi cho Trắng tại u, ta chỉ xem xét cây trò chơi gốc u tới độ cao h nào đó. áp dụng thủ tục Minimax cho cây trò chơi gốc u, độ cao h và sử dụng giá trị của hàm đánh giá cho các lá của cây đó, chúng ta sẽ tìm được nước đi tốt cho Trắng tại u.

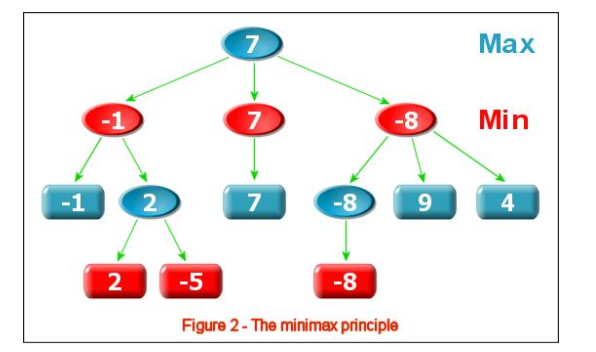
**3. Chiến lược cắt tỉa Alpha-Beta**

**3.1. Chiến lược MINIMAX**

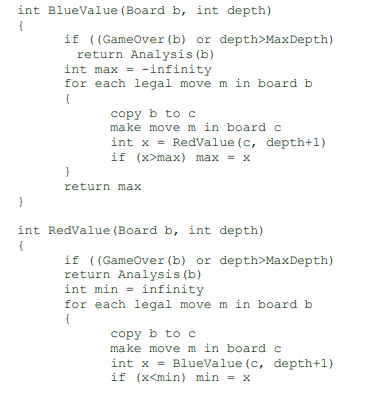
Nguyên lý: Nước đi tối ưu dành cho v, là nước đi duyệt qua tất cả các đỉnh con của u, là đỉnh tốt nhất trong số các đỉnh con của v.



Như vậy, vận dụng quy tắc trên, ta tìm hiểu hình bên dưới,

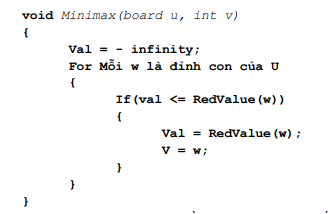


Với Blue (Quân Trắng), ta chọn Max(các children), Red (Quân Đen), ta tìm Min(các children). Từ hình vẽ trên, ta xây dựng các thủ tục đệ quy, gồm 2 hàm, BlueValue() đại diện cho Quân trắng, nhằm tìm nước đi tối ưu cho mình ( Max(children)), ngược lại, RedValue() đại diện cho quân đen, tìm Miin(Children) cho mình.





Trong đó, MaxDepth là độ sâu tối đa bạn duyệt đến, như vậy, duyệt càng sâu, bạn càng tiến gần đến thắng lợi. biến depth, được khởi tạo giá trị ban đầu là 0. Như vậy, tùy vào quân Trắng hay quân Đen mà bạn sẽ gọi hàm BlueValue() trước hay là RedValue() trước thông qua hàm Minimax() sau:



Trong đó, v là đỉnh mà ta cần tìm ( là đường đi tối ưu trong giới hạn nào đó). Như vậy, cho đến bây giờ bạn đã có thể xây dựng thành công trò chơi Dodgame. Nhưng để trò chơi được diễn ra nhanh hơn, ta cần có các hàm nhằm loại bỏ những nước đi không cần thiết, vui lòng bạn xem thuật toán AlphaBeta bên dưới.

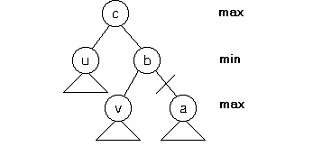
**3.2. Thuật toán Alpha-Beta**

Trong chiến lược tìm kiếm Minimax, để tìm kiếm nước đi tốt cho Trắng tại trạng thái u, cho dù ta hạn chế không gian tìm kiếm trong phạm vi cây trò chơi gốc u với độ cao h, thì số đỉnh cũng rất lớn với h >= 3.

Chẳng hạn, trong cờ vua, nhân tố nhánh trong cây trò chơi trung bình khoảng 35, thời gian đòi hỏi phải đưa ra nước đi là 150 giây, với thời gian này trên máy tính thông thường chương trình của bạn chỉ có thể xem xét các đỉnh trong độ sâu 3 hoặc 4.

Một người chơi cờ tr.nh độ trung bình cũng có thể tính trước được 5, 6 nước hoặc hơn nữa, và do đó chương trình của bạn mới đạt trình độ người mới tập chơi!

Khi đánh giá đỉnh u tới độ sâu h, một thuật toán Minimax đòi hỏi ta phải đánh giá tất cả các đỉnh của cây gốc u tới độ sâu h. Song ta có thể giảm bớt số đỉnh cần phải dánh giá mà vẫn không ảnh hưởng gì đến sự đánh giá đỉnh u. Phương pháp cắt cụt alpha-beta cho phép ta cắt bỏ các nhánh không cần thiết cho sự đánh giá đỉnh u.



Tư tưởng của kỹ thuật cắt cụt alpha-beta là như sau:

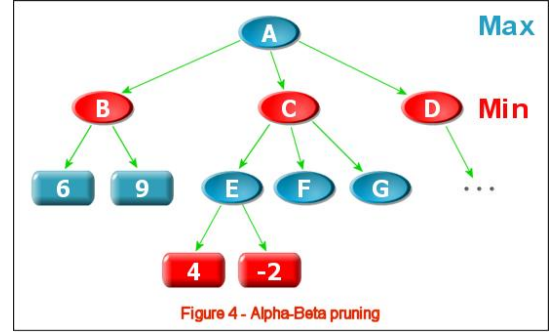
Nhớ lại rằng, chiến lược t.m kiếm Minimax là chiến lược tìm kiếm theo độ sâu. Giả sử trong quá trính tìm kiếm ta đi xuống đỉnh a là đỉnh Trắng, đỉnh a có người anh em v đã được đánh giá.

Giả sử cha của đỉnh a là b và b có người anh em u d. được đánh giá, và giả sử cha của b là c (Xem hình trên).

Khi đó ta có giá trị đỉnh c (đỉnh Trắng) ít nhất là giá trị của u, giá trị của đỉnh b (đỉnh Đen) nhiều nhất là giá trị v.

Do đó, nếu eval(u) > eval(v), ta không cần đi xuống để đánh giá đỉnh a nữa mà vẫn không ảnh hưởng gì dến đánh giá đỉnh c. Hay nói cách khác ta có thể cắt bỏ cây con gốc a. Lập luận tương tự cho trường hợp a là đỉnh Đen, trong trường hợp này nếu eval(u) < eval(v) ta cũng có thể cắt bỏ cây con gốc a.

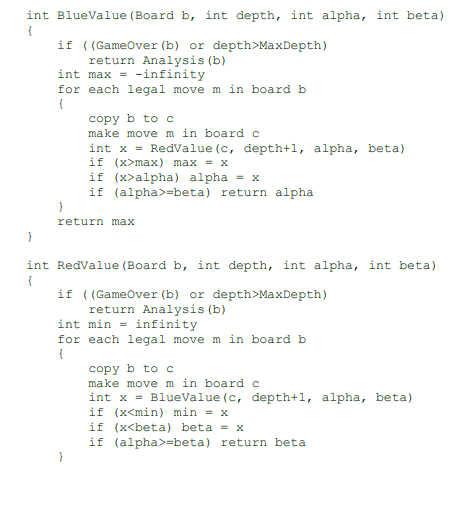
Cụ thể hơn, ta có hình bên dưới



Trong khi, ta đang cố tìm giá trị cho A, Chúng ta đi xuống B, và tìm được giá trị là 6 là Min(6,9). Sau đó, ta đi qua C. Từ đó, chúng ta tính được E là 4 là Max(4,-2). Do đó, C là nhỏ hơn,Ngay tại lúc này ta biết được giá trị của C là <=4, Do đó, C sẽ không được chọn bởi A là giá trị Maximum, vì A đã có B chứa giá trị 6, là giá trị chắc chắn lớn hơn bất kỳ giá trị nào của C (nếu ta tính tiếp). Do đó việc tìm giá tri Max cho F và G là không cần thiết nữa. Và lúc đó, ta sẽ dừng việc sử lý trên Node C, Và ta tiếp tục tìm trên Node D…

Như vậy, cho đến bây giờ ta có thể bổ sung kỹ thuật vừa nói trên vào thuật toán Minimax, ta có được chương trình hoàn thiện như sau:

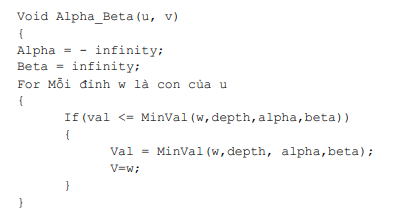
Để cài đặt kỹ thuật cắt cụt alpha-beta, đối với các đỉnh nằm trên đường đi từ gốc tới đỉnh hiện thời, ta sử dụng tham số a để ghi lại giá trị lớn nhất trong các giá trị của các đỉnh con để đánh giá của một đỉnh Trắng, còn tham số b ghi lại giá trị nhỏ nhất trong các đỉnh con để đánh giá của một đỉnh Đen. Giá trị của a và b sẽ được cập nhật trong quá trình tìm kiếm. a và b được sử dụng như các biến địa phương trong các hàm MaxVal(u, a, b) (hàm xác định giá trị của đỉnh Trắng u) và Minval(u, a, b) (hàm xác định giá trị của đỉnh Đen u).





Lưu Ý: Chúng ta viết alpha>= beta thay vì alpha>beta, vì điều này làm tăng tốc độ cho chương trình. Các hàm này được khởi tạo các giá trị ban đầu như sau: depth=0; alpha=-infinity, beta=infinity.

Như vậy, tùy vào quân Trắng hay quân Đen mà bạn sẽ gọi hàm BlueValue() trước hay là RedValue() trước thông qua hàm AlphaBeta() sau:



Trong đó, v là đường đi tối ưu cần tìm

**3.4. Các trường hợp cắt tỉa**

3.4.1. Cắt tỉa Alpha

Trường hợp cắt tỉa Alpha trên một nút Max . Giả sử ước đã xác định giá trị tốt nhất trên nhánh 1 có Min = 15 như hình vẽ. Ta bắt đầu tìm kiếm trên nhánh 2, đầu tiên ta tìm trên nhánh 2.1 giả sử ta nhận được kết quả là Min = 10 < 15 thì ta không cần tìm kiếm tiếp trên nhánh 2.2, vì giá của 2 Min ít nhất bằng 10, xét Max tại nút gốc thì Max ít nhất bằng 15 > 10. Do đó không cần xét các nhánh con còn lại của 2 nữa.

3.4.2. Cắt tỉa Beta

Trường hợp cắt tỉa Beta trên một nút Min. Giả sử kết thúc quá trình tìm kiếm trên nút 1.1 đạt Max = 10. Bắt đầu tìm kiếm trên nhánh 1.2, đầu tiên ta tìm trên nhánh 1.2.1 giá của nút 1.2.1 là 15, vậy Max tại nút 1.2 ít nhất bằng 15 > 10. Khi lấy Min tại nút 1 thì Min <= 10. Như vậy có thể cắt bỏ các nhánh con của 1.2 từ 1.2.2 trở đi.

**CHƯƠNG II. XÂY DỰNG CHƯƠNG TRÌNH**

**1. Thủ tục chính chương trình**

**1.1. Lớp Cell:**

Lớp Cell là một dẫn của lớp PictureBox, có các hàm chính:

– Hàm Public Cell (char type, int x, int y ): phương thức khởi dựng

– Hàm Void State (char type): biểu diễn trạng thái của một ô bàn cờ.

**1.2. Lớp Grid**

Lớp Grid gồm các thủ tục chính sau:

– Void PanelClick(object sender, MouseEventArgs e): xử lý sự kiên để đưa 1 quân cờ ra ngoài.

– Void CellClick(int x, int y): xử lý sự kiện khi click vào 1 ô trên bàn cờ.

– Void DisableOthers(): chuyển trạng thái của 1 ô sau 1 nước đi.

– Void UpdateCell(): cập nhật bàn cờ.

– Public Grid(int N): phương thưc khởi tạo bàn cờ, cho điểm các ô.

– Void DrawCell(): vẽ ô.

– Public GetState(): lấy trạng thái của bàn cờ.

– Public SetState(): thiết lập trạng thái của bàn cờ.

**1.3. Lớp Computer**

Lớp Computer gồm các thủ tục chính sau:

– Int MinVal(int dept, int alpha, int beta): hàm tính điểm cho MIN theo thuật toán cắt tỉa Alpha Beta.

– Int MaxVal(int dept, itn alpha, int beta ): hàm tính điểm cho MAX theo thuật toán cắt tỉa Alpha Beta.

– Void Getstate(): lấy trạng thái của bàn cờ.

– Void Setstate(): thiết lập trạng thái của bàn cờ.

– Byte [] Solve(byte[] state): hàm máy tính dùng để tính toán nước đi tiếp theo.

– Int Getpoint(): tính điểm tại 1 trạng thái của bàn cờ.

**1.4. Lớp GameManager**

Lớp GameManager gồm các thủ tục chính sau:

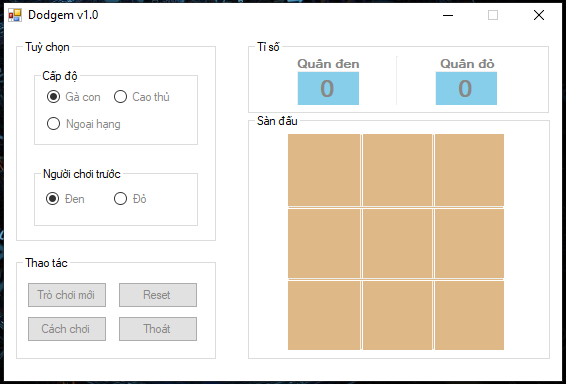
– Void newGame(): chọn trò chơi mới.

– Void SwitchPlayer(): đổi lượt chơi.

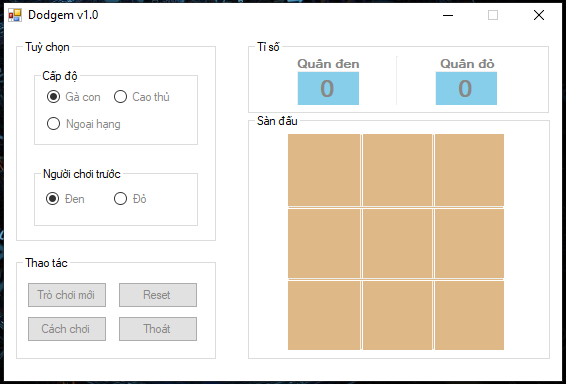
– Void Check(): kiểm tra xem trò chơi đã kết thúc chưa, xem ai là người thắng.

**2. Giao diện chương trình**

**2.1. Giao diện ban đầu**

****

**2.2. Giao diện tạo ván chơi mới**



**KẾT LUẬN**

Thuật toán Alpha-Beta nói chung giúp chúng ta tiết kiệm nhiều thời gian so với Minimax mà vẫn đảm bảo kết quả tìm kiếm chính xác. Tuy nhiên, lượng tiết kiệm này không ổn định – phụ thuộc vào số nút mà nó cắt bỏ. Trong trường hợp xấu nhất thuật toán không cắt được một nhánh nào và phải xét số nút đúng bằng Minimax. Ta cần đẩy mạnh việc cắt bỏ nhờ đẩy nhanh sự thu hẹp của cửa sổ tìm kiếm alpha – beta. Cửa sổ này được thu hẹp một bước khi gặp một giá trị mới tốt hơn giá trị cũ. Khi gặp giá trị tốt nhất thì cửa sổ này thu hẹp nhất. Càng sớm gặp giá trị tốt nhất thì cửa sổ càng chóng thu hẹp. Như vậy phải làm sao cho các nút ở lá được sắp xếp theo trật tự từ cao xuống thấp. Trật tự này càng tốt bao nhiêu thì thuật toán chạy càng nhanh bấy nhiêu.

Kết quả bài tập lớn đạt được:

– Hiểu được cách tìm có đối thủ, dạng tìm kiếm có chiều sâu cho bài toán trò chơi Dodgem.

– Sử dụng kỹ thuật cắt tỉa Alpha-Beta và cài đặt kỹ thuật này cho bài toán trò chơi Dodgem.